<u>CLAIMS</u>

Allowed Claims:

1. (Previously Presented) A software architecture embodied on one or more computer-readable storage media, the software architecture executed by a computing device for a distributed computing system comprising:

an application configured to handle requests submitted by remote devices over a network; and

a multi-tiered framework comprising:

an application program interface layer organized into multiple root namespaces, the application program interface layer to present functions used by the application to access network and computing resources of the distributed computing system;

a common language runtime layer, wherein calls to the application program interface layer are handed to the common language runtime layer supporting applications written in a plurality of different languages and translated into an intermediate supported language, the application program interface layer comprising various types related to constructing user interfaces, wherein the types define a collection of classes, interfaces, delegates, enumerations, and structures which belong to a group assigned a group name associated with one of the root namespaces, and wherein each of the types is referenced by a hierarchical name comprising a top level identifier prefixed to the group name;

an operating system layer or an object model service, wherein the calls handed to the common language runtime layer are executed locally by the operating system layer or the object model service; and

a common language specification for local execution by the operating system layer or the object model service, wherein the common language specification provides an ability to use a particular code module written in a first programming language with a code module written in a second programming language.

2. (Canceled)

3. (Original) A software architecture as recited in claim 1, wherein the distributed computing system comprises client devices and server devices that handle requests from the client devices, the remote devices comprising at least one client device.

4. (Original) A software architecture as recited in claim 1, wherein the distributed computing system comprises client devices and server devices that handle requests from the client devices, the remote devices comprising at least one server device that is configured as a Web server.

5. (Previously Presented) A multi-tiered architecture including an application program interface layer embodied on one or more computer readable storage media, comprising:

multiple types related to constructing user interfaces;

individual types being associated with one or more groups and being referenced by one or more hierarchical names, wherein each hierarchical name includes a top level identifier prefixed to a group name assigned to one of the one or more groups, the types comprising:

classes which represent managed heap allocated data that has reference assignment semantics;

interfaces that define a contract that other types can implement;

delegates that are object oriented function pointers; and

structures that represent static allocated data that has value assignment semantics and enumerations which are value types that represent named constants;

wherein the application program interface layer is associated with:

a common language runtime layer supporting applications written in a plurality of different languages and translated into an intermediate language supported by the common runtime layer; and

a common language specification for local execution by an operating system or an object model service, wherein the common language specification provides the ability to use a particular code module written in a first programming language with a code module written in a second programming language.

6. **(Original)** An application program interface as recited in claim 5, wherein the classes comprise a forms class that represents a window or a dialog box that makes up an application's user interface.

7. **(Original)** An application program interface as recited in claim 6, wherein the forms class has multiple members comprising one or more of: public static properties, public static methods, public instance constructors, public instance methods, public instance properties, public instance events, protected instance properties, and protected instance methods.

8. **(Original)** An application program interface as recited in claim 5, wherein the type comprising the interfaces comprises a button control interface that allows a control to act like a button on a form.

9. **(Original)** An application program interface as recited in claim 5, wherein the type comprising the interfaces comprises a container control interface that provides functionality for a control to act as a parent for other controls.

10. **(Original)** An application program interface as recited in claim 5, wherein the type comprising the interfaces comprises an editing notification interface.

11. **(Original)** An application program interface as recited in claim 5, wherein the type comprising the interfaces comprises a data object interface that provides a format independent mechanism for transferring data.

12. **(Original)** An application program interface as recited in claim 5, wherein the type comprising the interfaces comprises a feature support interface that specifies a standard interface for retrieving feature information from a current system.

13. **(Original)** An application program interface as recited in claim 5, wherein the type comprising the interfaces comprises a message filter interface.

14. **(Original)** An application program interface as recited in claim 5, wherein the type comprising the interfaces comprises a handle-exposing interface to expose handles.

15. **(Original)** An application program interface as recited in claim 5, wherein the type comprising the interfaces comprises one or more of the following interfaces:

a button control interface that allows a control to act like a button on a form;

a container control interface that provides functionality for a control to act as a parent for other controls;

an editing notification interface;

a data object interface that provides a format independent mechanism for transferring data;

a feature support interface that specifies a standard interface for retrieving feature information from a current system;

a message filter interface; and

a handle-exposing interface to expose handles.

16. **(Currently Amended)** A distributed computer software architecture embodied on one or more computer-readable storage media, the distributed computer software architecture comprising:

one or more applications configured to be executed on one or more computing devices, the applications handling requests submitted from remote computing devices;

a networking platform to support the one or more applications;

an application programming interface to interface the one or more applications with the networking platform, the application programming interface comprising various types related to constructing user interfaces, individual types being associated with one or more groups and being referenced by one or more hierarchical names, wherein each of the hierarchical names includes a top level identifier prefixed to a group name assigned to one of the one or more groups;

a common language runtime layer supporting applications written in a plurality of different languages and translated into an intermediate language supported by the common runtime layer ~~and a common language specification for local execution by an operating system or an object model service~~; and

a common language specification for local execution by [[the]] <u>an</u> operating system or [[the]] <u>an</u> object model service, wherein the common language specification provides an ability to use a particular code module written in a first programming language with a code module written in a second programming language.

17. (Canceled)

18. (Previously Presented) A distributed computer software architecture as recited in claim 42, wherein the classes comprises a forms class that represents a window or a dialog box that makes up an application's user interface.

19. (Original) A distributed computer software architecture as recited in claim 18, wherein the forms class has multiple members comprising one or more of: public static properties, public static methods, public instance constructors, public instance methods, public instance properties, public instance events, protected instance properties, and protected instance methods.

20. (Previously Presented) A distributed computer software architecture as recited in claim 42, wherein the type comprising the interfaces comprises a button control interface that allows a control to act like a button on a form.

21. (Previously Presented) A distributed computer software architecture as recited in claim 42, wherein the type comprising the interfaces comprises a container control interface that provides functionality for a control to act as a parent for other controls.

22. **(Previously Presented)** A distributed computer software architecture as recited in claim 42, wherein the type comprising the interfaces comprises an editing notification interface.

23. **(Previously Presented)** A distributed computer software architecture as recited in claim 42, wherein the type comprising the interfaces comprises a data object interface that provides a format independent mechanism for transferring data.

24. **(Previously Presented)** A distributed computer software architecture as recited in claim 42, wherein the type comprising the interfaces comprises a feature support interface that specifies a standard interface for retrieving feature information from a current system.

25. **(Previously Presented)** A distributed computer software architecture as recited in claim 42, wherein the type comprising the interfaces comprises a message filter interface.

26. **(Previously Presented)** A distributed computer software architecture as recited in claim 42, wherein the type comprising the interfaces comprises a handle-exposing interface to expose handles.

27. (Previously Presented) A distributed computer software architecture as recited in claim 42, wherein the type comprising the interfaces comprises one or more of the following interfaces:

a button control interface that allows a control to act like a button on a form;

a container control interface that provides functionality for a control to act as a parent for other controls;

an editing notification interface;

a data object interface that provides a format independent mechanism for transferring data;

a feature support interface that specifies a standard interface for retrieving feature information from a current system;

a message filter interface; and

a handle-exposing interface to expose handles.

28. (Previously Presented) A computer system including one or more microprocessors and one or more software programs stored on one or more computer-readable storage media, the one or more software programs utilizing an application program interface to request services from an operating system, the application program interface including separate commands to request services comprising services related to constructing user interfaces, wherein the application program interface groups API functions into multiple namespaces that define a collection of classes which represent managed heap allocated data that has reference assignment semantics, interfaces that define a contract that other types can implement, delegates that are object oriented function pointers, enumerations which are value types that represent named constants and structures that represent static allocated data that has value assignment semantics, the application program interface being associated with:

a common language runtime layer supporting applications written in a plurality of different languages and translated into an intermediate language supported by the common runtime layer; and

a common language specification for local execution by an operating system or an object model service, wherein the common language specification provides an ability to use a particular code module written in a first programming language with a code module written in a second programming language; and

wherein the type comprising the interfaces comprises one or more of the following interfaces:

a button control interface that allows a control to act like a button on a form;

a container control interface that provides functionality for a control to act as a parent for other controls;

an editing notification interface;

a data object interface that provides a format independent mechanism for transferring data;

a feature support interface that specifies a standard interface for retrieving feature information from a current system;

a message filter interface; or

a handle-exposing interface to expose handles.

29. **(Previously Presented)** A method, comprising:

managing network and computing resources for a distributed computing system;

exposing a set of functions that enable developers to access the network and computing resources of the distributed computing system, the set of functions comprising functions to facilitate construction of user interfaces, wherein the user interfaces include windowing, menus, and dialogs, and wherein the functions are grouped into multiple namespaces that define a collection of classes which represent managed heap allocated data that has reference assignment semantics, interfaces that define a contract that other types can implement, delegates that are object oriented function pointers, enumerations which are value types that represent named constants and structures that represent static allocated data that has value assignment semantics;

using a common language runtime layer supporting applications written in a plurality of different languages and translated into an intermediate language supported by the common runtime layer; and

using a common language specification for local execution by an operating system or an object model service, wherein the common language specification provides an ability to use a particular code module written in a first programming language with a code module written in a second programming language.

30. **(Original)** A method as recited in claim 29, further comprising receiving a request from a remote computing device, the request containing a call to the set of functions.

31-40. (Canceled)

41. (Previously Presented) A software architecture as recited in claim 1, wherein the various types comprise classes, interfaces, delegates, structures and enumerations.

42. (Previously Presented) A distributed computer software architecture as recited in claim 16, wherein the various types comprise classes, interfaces, delegates, structures and enumerations.